CLAIMS

We claim:

1. An electronic design automation system for verifying a user design, comprising:

a computing system including a central processing unit for modeling the user design in software;

an internal bus system coupled to the computing system;

reconfigurable hardware logic coupled to the internal bus system and for generating a hardware model which includes at least a portion of the user design modeled in hardware; and

control logic coupled to the internal bus system for controlling the delivery of data between the reconfigurable hardware logic and the computing system; and

shared memory for holding a first information of the software model and a second information of the hardware model.

2. The system of claim 1, wherein the first information of the software model includes functional information of the user design.

3. The system of claim 1, wherein the second information of the hardware model includes functional information of the user design.

4. The system of claim 2, wherein the second information of the hardware model includes state information of the user design in the reconfigurable hardware logic.

5. The system of claim 1, wherein the control logic includes a direct memory access (DMA) engine for loading the second information of the hardware model to the shared memory.

6. The system of claim 5, wherein the second information of the hardware model includes state information of the user design in the reconfigurable hardware logic.

7. The system of claim 1, further comprising:

timing logic associated with each latch so that a desired result is obtained regardless of the order of arrival of a data input and a clock signal to the latch.

8. The system of claim 7, wherein the timing logic further comprises:

a first logic having a first input for receiving a first data at a first value, a second input, a first output, and a control input for receiving a clock signal; and

a second logic for storing a current value having a first trigger input, a second logic input coupled to the first output, and a second logic output coupled to the second input of the first logic, wherein the second logic updates to the first value of the first data that is a correct value at the first output and presents the first data to the second input of the first logic when a trigger signal is received at the trigger input, regardless of the order of arrival of the clock signal at the control input or the first input to the first logic.

9. The system of claim 8, further comprising:

a third logic for storing a new value and having a third logic input, a second trigger input, and a third logic output, where the third logic output is coupled to the first input of the first logic; and

an edge detector having a clock input, a third trigger input, and an edge detector output, where the edge detector output is coupled to the control input, wherein the trigger signal is applied to the second and third trigger inputs at selected times to update the logic apparatus.

10. The system of claim 1, further comprising:

timing logic associated with each flip-flop so that a desired result is obtained regardless of

the order of arrival of a data input and a clock signal to the flip-flop.

11. The system of claim 10, wherein the timing logic further comprises:

an input logic for receiving a new input value and a trigger signal;

a storage logic for storing an old input value and the trigger signal;

a transmission logic coupled to the input logic for receiving the new input value and coupled to the storage logic for receiving the old input value and selecting from one of the new input value and the old input value to generate a flip-flop output; and

an edge detecting logic for detecting an edge of a clock signal and receiving the trigger signal, wherein the transmission logic outputs the new input value upon the reception of the trigger signal .

12. The flip-flop of claim 11, wherein the storage logic receives the new input value and stores the new input value upon the reception of the trigger signal.

13. The flip-flop of claim 11, wherein the edge detecting logic is a positive-edge detecting logic for detecting a positive edge of the clock signal.

14. The flip-flop of claim 13, wherein the edge detecting logic is coupled to the transmission logic and generates a selector signal to the transmission logic for selecting either the new input value or the old input value.

15. The flip-flop of claim 11, wherein the input logic is a D flip-flop which receives the new input value and having a clock input for receiving the trigger signal.

16. The flip-flop of claim 11, wherein the storage logic is a D flip-flop whose input is coupled to

the flip-flop output of the transmission logic and having a clock input for receiving the trigger signal.

17. The flip-flop of claim 11, wherein the transmission logic further includes:

a multiplexer for receiving the new input value and the old input value, and having a selector input for receiving a selector signal from the edge detecting logic, and a mux output,

an OR gate for receiving the mux output and a set input, and having an OR gate output, and

an AND gate for receiving the OR gate output and a reset input, and providing the flip-flop output.

18. The flip-flop of claim 11, wherein the edge detecting logic includes:

a D flip-flop for receiving a clock signal and a clock input for receiving the trigger signal, and having a D flip-flop output,

an AND gate for receiving the clock signal and the D flip-flop output, and having a selector output for providing a selector signal.

19. A method of simulating a circuit in a simulation system, the circuit having a structure and a function specified in a hardware language, the hardware language capable of describing the circuit as component types and connections, comprising steps:

determining component type in the hardware language;

generating a software model of the circuit;

generating a hardware model of at least a portion of the circuit based on component type automatically;

allocating space in a shared memory for the software model and the hardware model; and

simulating the behavior of the circuit with the software model by initially using the state information in the shared memory.

20. The method of claim 19, further comprising step:

loading state information of the hardware model via direct memory access (DMA).

21. The method of claim 19, further comprising step:

controlling the software model and the hardware model with a software kernel.

22. The method of claim 21, wherein the step of controlling further comprises steps:

determining the presence of input data to the simulation system;

evaluating clock components;

propagating input data to the hardware model;

detecting active clock edge of the clock components in the software model; and

evaluating the input data with the hardware model in response to the active clock edge detection.

23. The method of claim 19, wherein the step of simulating further comprises:

simulating the behavior of the circuit with the software model for a time period; and

simulating the behavior of the circuit with the hardware model for another time period to accelerate the simulation process.

24. A method of simulating a circuit, the circuit having a structure and a function specified in a hardware language, the hardware language capable of describing the circuit as component types and connections, comprising steps:

generating a software model of the circuit;

generating a hardware model of the circuit;

allocating space in a shared memory for the software model and the hardware model;

simulating a behavior of the circuit with the software model by providing input data to the software model;

selectively switching to the hardware model through software control;

providing input data to the hardware model; and

evaluating the input data in the hardware model based on the detection of a trigger event in the software model.

25. The method of claim 24, further comprising step:

loading the shared memory with state information from the hardware model.

26. The method of claim 25, further comprising step:

simulating the behavior of the client with the software model by initially using state information from the hardware model in shared memory.

27. The method of claim 24, wherein the step of generating the hardware model further comprises steps:

determining component type in the hardware language; and

generating the hardware model based on component type.

28. The method of claim 24, further comprising steps:

selectively switching to the software model; and

simulating a behavior of the circuit with the software model by providing input data to the software model.

29. The method of claim 24, wherein the step of evaluating further comprises:

determining the presence of input data to the simulation system;

evaluating clock components;

propagating input data to the hardware model;

detecting the trigger event, wherein the trigger event includes an active clock edge of the clock components; and

evaluating the input data with the hardware model in response to the active clock edge detection.

30. A method of evaluating data in a circuit during a simulation process, comprising:

generating a software model of the circuit;

generating a hardware model of at least a portion of the circuit;

allocating space in shared memory for the software model and the hardware model;

propagating data to the hardware model until the data stabilizes;

detecting a clock edge in the software model; and

evaluating data with the hardware model in response to the clock edge detection in the software model and in synchronization with a hardware-generated clock.

31. The method of claim 30, further comprising step:

loading the shared memory with state information from the hardware model.

32. The method of claim 31, further comprising step:

simulating the circuit with the software model by initially using the state information from the hardware model.

33. A simulation system operating in a host computer system for simulating a behavior of a circuit, the host computer system including a central processing unit (CPU), shared memory, and a local bus coupling the CPU to main memory and allowing communication between the CPU and main memory, the circuit having a structure and a function specified in a hardware language, the hardware language capable of describing the circuit as component types and connections, comprising:

a software model of the circuit coupled to the local bus;

software control logic coupled to the software model and a hardware logic element, for controlling the operation of the software model and said hardware logic element;

said hardware logic element coupled to the local bus and including a hardware model of at least a portion of the circuit configured automatically based on component type; and

DMA engine for loading state information of the hardware model from the hardware logic element to the shared memory.

34. The system of claim 33, wherein the software control logic further comprises:

interface logic which is capable of receiving input data and a clock data from an external process, and

clock detection logic for detecting an active edge of the clock data and generating a trigger signal.

35. The system of claim 34, wherein the hardware logic element further comprises:

clock enable logic for evaluating data in the hardware model in response to the trigger signal.

36. The system of claim 33, wherein the hardware logic element comprises a field programmable device.

37. The system of claim 33, wherein the hardware logic element comprises:

a plurality of field programmable devices coupled together, each field programmable device including a portion of the hardware model of the circuit;

a plurality of interconnections to couple the portions of the hardware model together, each interconnection representing a direct connection between field programmable devices, wherein the shortest path between any two field programmable devices is at most two interconnections.

38. A coverification system for verifying a user design, comprising:

a computing system including a central processing unit and memory for modeling the user design in software;

an internal bus system coupled to the computing system;

reconfigurable hardware logic coupled to the internal bus system and for modeling at least a portion of the user design in hardware;

an external interface coupled to the internal bus system and at least one external device;

control logic coupled to the internal bus system for controlling the delivery of data among the reconfigurable hardware logic, the computing system, and the external interface; and

shared memory for loading state information of the user design from the reconfigurable hardware logic to the computing system.

39. The coverification system of claim 38, wherein the control logic further comprises:

a data-in control logic for controlling delivery of data from the computing system and the external interface to the reconfigurable hardware logic, including

data-in pointer logic coupled to the internal bus system for generating selective pointer signals to a data-in latch logic, the generation of selective pointer signals based on whether the data is arriving from the computing system or the external interface and the particular internal nodes in the reconfigurable hardware logic selected to be driven, and

data-in latch logic coupled to the internal bus system, a plurality of internal nodes in the reconfigurable hardware logic, and the data-in pointer logic for delivering data from the internal bus system to selective internal nodes in the reconfigurable hardware logic in response to the selective pointer signals.

40. The coverification system of claim 39, further comprising:

an external buffer coupled to the external interface for storing data originating from the external interface and also coupled to the internal bus system the computing system has access to data in the external buffer.

41. The coverification system of claim 39, wherein the control logic further comprises:

a data-out control logic for controlling delivery of data from the reconfigurable hardware logic to the computing system and the external interface, including,

data-out pointer logic coupled to the internal bus system for generating selective pointer signals to a data-out gating logic, the generation of selective pointer signals based on whether the data is destined for the computing system or the external interface and the particular internal nodes in the reconfigurable hardware logic selected to be driven, and

data-out gating logic coupled to the internal bus system, a plurality of internal nodes in the reconfigurable hardware logic, and the data-out pointer logic for delivering data from the selective internal nodes in the reconfigurable hardware logic to the internal bus system in response to the selective pointer signals.

42. The coverification system of claim 38, further comprising:

software clock logic for detecting an active clock edge of a clock signal in the software model during data evaluation, generating a software clock to the reconfigurable hardware logic to control a corresponding data evaluation in the reconfigurable hardware logic, and generating the software clock to the external interface.

43. The coverification system of claim 38, wherein the computing system further includes at least one model of an external I/O device in software.

44. A method of verifying the proper operation of a user design connected to an external I/O device, comprising steps:

generating a first model of the user design in software;

loading the first model in a shared memory;

generating a second model of at least a portion of the user design in hardware;

controlling the second model in hardware with the first model in the software; and

loading state information from the second model to the shared memory.

45. The method of claim 44, wherein the step of controlling further comprises:

synchronizing the data evaluation in the first model in software and the second model in hardware with a software-generated clock.

46. The method of claim 45, further comprising steps:

simulating selected debug test points in software;

accelerating selected debug test points in hardware; and

controlling the delivery of data among the first model in software, the second model in hardware, and the external I/O device so that the first model in software has access to all delivered data.

47. The method of claim 46, wherein the step of controlling further comprising steps:

selecting at least one internal node in the reconfigurable hardware logic;

determining if the data being delivered is from the first model in software or the external I/O device; and

generating selected pointer signals to at least one latching logic coupled to the selected internal node based on the selecting and determining steps so that the data is delivered from either the first model in software or the external I/O device to the second model in hardware.

48. The method of claim 47, further comprising steps:

storing data delivered from the external I/O device in an external buffer coupled to the second model in hardware; and

providing the first model in software access to data in the external buffer.

49. The method of claim 46, wherein the step of controlling further comprising steps:

selecting at least one internal node in the reconfigurable hardware logic;

determining if the data being delivered is destined for the first model in software or the external I/O device; and

generating selected pointer signals to at least one gating logic coupled to the selected internal node based on the selecting and determining steps so that the data is delivered from the second model in hardware to either the first model in software or to both the first model in software and the external I/O device.

50. The method of claim 44, further comprising step:

generating a model of an external I/O device in software.